ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ имени академика С.П. КОРОЛЁВА»

КАФЕДРА «ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ

«Объектно-ориентированное программирование»

ЛАБОРАТОРНАЯ РАБОТА №3

Студент Луковникова Е.Д

Группа 6301-030301D

Руководитель Борисов Д. С.

Оценка _____

САМАРА 2025

Задание №1

1) java.lang.Exception - является корневым классом для всех некритических ошибок, которые программа может (и должна) попытаться обработать.

2) java.lang.IndexOutOfBoundsException – исключение происходит тогда, когда программа пыталась получить доступ к элементу по индексу, который выходит за допустимые границы

3) java.lang.ArrayIndexOutOfBoundsException – частный случай для второго исключения, когда работа идет с массивом, то есть была попытка получить элемент по индексу, который выходит за границы массива.

4) java.lang.IllegalArgumentException - Указывает на то, что метод получил аргумент, который является недопустимым, даже если он синтаксически правилен. Это значит, что значение аргумента не соответствует требованиям, предъявляемым методом.

5) java.lang.IllegalStateException - Указывает на то, что метод был вызван в неподходящее время, или что объект находится в недопустимом состоянии для выполнения данного метода. Операция не может быть выполнена из-за текущего состояния объекта.

Задание №2

Созданы специализированные классы исключений для обработки ошибочных ситуаций. Исключение для индексов, выходящих за границы, позволяет через метод getMessage() в блоке try-catch выводить детальное описание причины ошибки в основном классе Main.

```java
package functions;
public class FunctionPointIndexOutOfBoundsException extends IndexOutOfBoundsException {
    public FunctionPointIndexOutOfBoundsException() {
        super();
    }

    public FunctionPointIndexOutOfBoundsException(String message) {
        super(message);
    }
}
```

Созданы исключения для обработки различных ошибочных ситуаций в программе. Использование try-catch в Main с методом getMessage() позволяет выводить детальные описания причин возникновения исключений.

```java
package functions;

public class InappropriateFunctionPointException extends Exception {
    public InappropriateFunctionPointException() {
        super();
    }

    public InappropriateFunctionPointException(String message) {
        super(message);
    }
}
```

Задание №3

В разработанный ранее класс TabulatedFunction внесены изменения, обеспечивающие выбрасывание исключений методами класса.

- Оба конструктора класса выбрасывают исключение IllegalArgumentException, если левая граница области определения больше или равна правой, а также если предлагаемое количество точек меньше двух. Это обеспечивает создание объекта только при корректных параметрах.
- Методы getPoint(), setPoint(), getPointX(), setPointX(), getPointY(), setPointY() и deletePoint() выбрасывают исключение FunctionPointIndexOutOfBoundsException, если переданный в метод номер выходит за границы набора точек. Это обеспечивает корректность обращений к точкам функции.
- Методы setPoint() и setPointX() выбрасывают исключение InappropriateFunctionPointException в том случае, если координата x задаваемой точки лежит вне интервала, определяемого значениями соседних точек табулированной функции. Метод addPoint() выбрасывает исключение InappropriateFunctionPointException, если в наборе точек функции есть точка, абсцисса которой совпадает с абсциссой добавляемой точки. Это обеспечивает сохранение упорядоченности точек функции.
- Метод deletePoint() выбрасывает исключение IllegalStateException, если на момент удаления точки количество точек в наборе менее трех. Это обеспечивает невозможность получения функции с некорректным количеством точек.

```
public ArrayTabulatedFunction(double leftX, double rightX, int pointsCount) {
    if (pointsCount < 3){
        throw new IllegalArgumentException(s: "Количество точек должно быть не менее 3");
    }

    if (leftX>=rightX){
        throw new IllegalArgumentException(s: "Правая граница должна быть больше левой");
    }
```

```
public ArrayTabulatedFunction(double leftX, double rightX, double[] values){
    if (values == null){
        throw new IllegalArgumentException(s: "Массив не может быть пустым");
    }

    if (values.length < 2){
        throw new IllegalArgumentException(s: "Массив должен содержать не менее 2");
    }

    if (leftX>=rightX){
        throw new IllegalArgumentException(s: "Правая граница должна быть больше левой");
    }
```

```
public FunctionPoint getPoint(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
    return new FunctionPoint(points[index]);
}
```

```java
public void setPoint(int index, FunctionPoint point) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
```

```java
public double getPointX(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
```

```java
public void setPointX(int index, double x) throws InappropriateFunctionPointException{

    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }


    if (x <= points[index - 1].getX()) {
        throw new InappropriateFunctionPointException();
    }
    if (x >= points[index + 1].getX()) {
        throw new InappropriateFunctionPointException();
    }
```

```java
public double getPointY(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
```

```java
public void setPointY(int index, double y) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
```

```java
public void deletePoint(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new IndexOutOfBoundsException(s: "Индекс выходит за границы");
    }
    if (pointsCount < 3) {
        throw new IllegalStateException(s: "Невозможно удалить - нужно 3 точки");
    }
```

```java
if (insertIndex < pointsCount && points[insertIndex].getX() == point.getX()) {
    throw new InappropriateFunctionPointException();
}
```

Задание №4

Для работы со связным списком был создан класс, имеющий в себе конструктор, позволяющий создать новый узел в списке, методы, позволяющие получить или изменить точку в узле, получить или изменить ссылку на предыдущий узел и получить или изменить ссылку на следующий узел.

```
package functions;

public class LinkedListTabulatedFunction implements TabulatedFunction{

    private class FunctionNode {
        private FunctionPoint point;
        private FunctionNode pred;
        private FunctionNode next;

        public FunctionNode(FunctionPoint point, FunctionNode pred, FunctionNode next) {
            this.point = point;
            this.pred = pred;
            this.next = next;
        }

        public FunctionPoint getPoint() {
            return point;
        }

        public void setPoint(FunctionPoint point) {
            this.point = point;
        }

        public FunctionNode getPred() {
            return pred;
        }

        public void setPred(FunctionNode prev) {
            this.pred = prev;
        }

        public FunctionNode getNext() {
            return next;
        }

        public void setNext(FunctionNode next) {
            this.next = next;
        }
    }
```

Метод позволяющий получить точку, находящуюся на переданном индексе, для оптимизации поиска, метод находит к какой границе ближе находится точка и находит ее смещением ссылки.

```
private FunctionNode head;
private int pointsCount;
private final double EPSILON_DOUBLE = 1e-9;

private FunctionNode getNodeByIndex(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }

    FunctionNode current;

    if (index < pointsCount / 2) {
        current = head.getNext();
        for (int i = 0; i < index; i++) {
            current = current.getNext();
        }
    } else {
        current = head;
        for (int i = pointsCount; i > index; i--) {
            current = current.getPred();
        }
    }

    return current;
}
```

Метод позволяющий добавить элемент в конец списка

```
private FunctionNode addNodeToTail(FunctionPoint point) {
    FunctionNode newNode = new FunctionNode(point, head.getPred(), head);
    FunctionNode tail = head.getPred();
    tail.setNext(newNode);
    head.setPred(newNode);
    return newNode;
}
```

Метод позволяющий добавить элемент на переданный индекс.

```
private FunctionNode addNodeByIndex(int index, FunctionPoint point) {
    if (index < 0 || index > pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    if (index == pointsCount) {
        pointsCount++;
        return addNodeToTail(point);
    }
    FunctionNode nextNode = getNodeByIndex(index);
    FunctionNode prevNode = nextNode.getPred();
    FunctionNode newNode = new FunctionNode(point, prevNode, nextNode);
    prevNode.setNext(newNode);
    nextNode.setPred(newNode);
    pointsCount++;
    return newNode;
}
```

И метод позволяющий удалить точку по переданному индексу.

```
private FunctionNode deleteNodeByIndex(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    FunctionNode nodeToDelete = getNodeByIndex(index);
    FunctionNode predNode = nodeToDelete.getPred();
    FunctionNode nextNode = nodeToDelete.getNext();
    predNode.setNext(nextNode);
    nextNode.setPred(predNode);
    pointsCount--;
    return nodeToDelete;
}
```

Два конструктора создающих список в двух случаях, когда задано количество точек или массив точек Y.

```java
public LinkedListTabulatedFunction(double leftX, double rightX, int pointsCount){
    this.pointsCount = pointsCount;
    if (leftX >= rightX) {
        throw new IllegalStateException(s: "В массиве не может быть только одна точка");
    }
    if (pointsCount < 3) {
        throw new IllegalStateException(s: "В массиве не может быть только одна точка");
    }
    head = new FunctionNode(point: null, pred: null, next: null);
    head.setNext(head);
    head.setPred(head);
    double step = (rightX - leftX)/(pointsCount - 1);
    for (int i = 0; i < pointsCount; i++) {
        addNodeToTail(new FunctionPoint(leftX + step * i, y: 0));
    }
}

public LinkedListTabulatedFunction(double leftX, double rightX, double[] values){
    this.pointsCount = values.length;
    if (leftX >= rightX) {
        throw new IllegalStateException(s: "В массиве не может быть только одна точка");
    }
    if (pointsCount < 3) {
        throw new IllegalStateException(s: "В массиве не может быть только одна точка");
    }
    head = new FunctionNode(point: null, pred: null, next: null);
    head.setNext(head);
    head.setPred(head);
    double step = (rightX - leftX)/(pointsCount - 1);
    for (int i = 0; i < pointsCount; i++) {
        addNodeToTail(new FunctionPoint(leftX + step * i,  values[i]));
    }
}
```

Методы, позволяющие получить левую и правую границы списка

```java
@Override
public double getLeftDomainBorder() {
    return head.getNext().getPoint().getX();
}

@Override
public double getRightDomainBorder() {
    return head.getPred().getPoint().getX();
}
```

Метод позволяющий найти координату Y с помощью линейной интерполяции для переданного X.

```java
public double getFunctionValue(double x) {
    if (x <  getLeftDomainBorder()|| x > getRightDomainBorder()) {
        return Double.NaN;
    }
    if (Math.abs(x - getLeftDomainBorder()) < EPSILON_DOUBLE) {
        return head.getNext().getPoint().getY();
    }
    if (Math.abs(x  - getRightDomainBorder()) < EPSILON_DOUBLE) {
        return head.getPred().getPoint().getY();
    }
else {
int i;
double value = 0;
for (i = 0; x >= getNodeByIndex(i).getPoint().getX(); i++){
FunctionNode Node = getNodeByIndex(i);
if (Math.abs(x - Node.getPoint().getX()) < EPSILON_DOUBLE){
    value = Node.getPoint().getY();
}
else{
value = Node.getPoint().getY() + (Node.getNext().getPoint().getY() - Node.getPoint().getY())*(x - Node.getPoint().getX())/(Node.getNext().getPoint().getX() - Node.getPoint().getX());
}
}
return value;
}
}
```

Методы для получения количества точек в списке и для получения и изменения точки в списке

```java
@Override
public int getPointsCount() {
    return pointsCount;
}

@Override
public FunctionPoint getPoint(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    return new FunctionPoint(getNodeByIndex(index).getPoint().getX(), getNodeByIndex(index).getPoint().getY());
}

@Override
public void setPoint(int index, FunctionPoint point) throws InappropriateFunctionPointException{
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    FunctionNode currentNode = getNodeByIndex(index);
    if (index == 0) {
        if (point.getX() > getRightDomainBorder()) {
            throw new InappropriateFunctionPointException("Новая точка X (" + point.getX() + ") выходит за границы (больше правой границы).");
        }
    } else if (index == pointsCount - 1) {
        if (point.getX() < getLeftDomainBorder()) {
            throw new InappropriateFunctionPointException("Новая точка X (" + point.getX() + ") выходит за границы (меньше левой границы).");
        }
    }
    else {
        if (point.getX() < currentNode.getPred().getPoint().getX() || point.getX() > currentNode.getNext().getPoint().getX()) {
            throw new InappropriateFunctionPointException("Новая точка X (" + point.getX() + ") выходит за границы соседних к ней точек.");
        }
    }
    currentNode.setPoint(point);
}
```

Геттер и сеттер методы для точек X и Y.

```java
@Override
public double getPointX(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    else{
        return getNodeByIndex(index).getPoint().getX();
    }
}

@Override
public void setPointX(int index, double x) throws InappropriateFunctionPointException {
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    FunctionNode currentNode = getNodeByIndex(index);
    if (x <  currentNode.getPred().getPoint().getX() || x > currentNode.getNext().getPoint().getX()) {
        throw new InappropriateFunctionPointException("Новая точка X (" + x + ") выходит за границы соседних к ней точек.");
    }
    currentNode.getPoint().setX(x);
}

@Override
public double getPointY(int index) {
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    else{
        return  getNodeByIndex(index).getPoint().getY();
    }
}

@Override
public void setPointY(int index, double y) {
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    else{
        FunctionNode currentNode = getNodeByIndex(index);
        currentNode.getPoint().setY(y);
    }
}

@Override
public void deletePoint(int index){
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    if (pointsCount < 3) {
        throw new IllegalStateException(s: "В массиве меньше 3 точек");
    }
    deleteNodeByIndex(index);
}
```

Методы для удаления и добавления точек в список

```java
@Override
public void deletePoint(int index){
    if (index < 0 || index >= pointsCount) {
        throw new FunctionPointIndexOutOfBoundsException(message: "Индекс выходит за границы");
    }
    if (pointsCount < 3) {
        throw new IllegalStateException(s: "В массиве меньше 3 точек");
    }
    deleteNodeByIndex(index);
}

@Override
public void addPoint(FunctionPoint point) throws InappropriateFunctionPointException {
    if (point.getX() > getNodeByIndex(pointsCount - 1).getPoint().getX()){
        addNodeByIndex(pointsCount, point);
    }
    else{
        int i = 0;
        while (point.getX() > getNodeByIndex(i).getPoint().getX()) ++i;
        FunctionNode Node = getNodeByIndex(i);
        if (Math.abs(point.getX() - Node.getPoint().getX()) < EPSILON_DOUBLE) {
            throw new InappropriateFunctionPointException("Координата X добавляемой точки совпадает с уже сужествующим X = " +  Node.getPoint().getX());
        }
        else{
            addNodeByIndex(i, point);
        }
    }
}
```

В методы и конструкторы, как и в ArrayTabulatedFunction добавлены выбрасывания исключений для необходимых случаев.

Задание №6

Класс TabulatedFunction переименован в ArrayTabulatedFunction. Создан интерфейс, который обеспечивает автоматическое определение рабочего класса - ArrayTabulatedFunction или LinkedListTabulatedFunction.

```java
package functions;

public interface TabulatedFunction{
    double getLeftDomainBorder();
    double getRightDomainBorder();
    int getPointsCount();
    FunctionPoint getPoint(int index);
    void setPoint(int index,FunctionPoint point) throws InappropriateFunctionPointException ;
    double getPointX(int index);
    void setPointX(int index,double x) throws InappropriateFunctionPointException;
    double getPointY(int index);
    void setPointY(int index,double y);
    double getFunctionValue(double x);
    void deletePoint(int index) throws InappropriateFunctionPointException;
    void addPoint(FunctionPoint point) throws InappropriateFunctionPointException;
}
```

Задание № 7

Реализована обработка исключений в основном классе: для всех методов, способных генерировать проверяемые исключения (addPoint и setPoint), добавлены соответствующие блоки try-catch, обеспечивающие корректное выполнение программы при возникновении ошибок.

Результат работы программы

```
Использование ArrayTabulatedFunction:
Исходная функция:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,0, 9,0)
Точка 4: (4,0, 16,0)

Значения функции в различных точках:
Значение функции X = 3.3 Y = 11.099999999999998

После изменения точек:
После изменения y в точке 3 на 8.0:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,0, 8,0)
Точка 4: (4,0, 16,0)
После изменения x в точке 3 на 3.5:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,5, 8,0)
Точка 4: (4,0, 16,0)

После добавления и удаления точек:
После добавления x в точке 3 на 3.5:
Область определения: [0,0, 4,0]
Количество точек: 6
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,0, 3,5)
Точка 4: (3,5, 8,0)
Точка 5: (4,0, 16,0)
```

```
После удаления точки с индексом 3:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,5, 8,0)
Точка 4: (4,0, 16,0)
Использование LinkenListTabulatedFunction:
Исходная функция:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,0, 9,0)
Точка 4: (4,0, 16,0)

Значения функции в различных точках:
Значение функции X = 3.3 Y = 11.099999999999998

После изменения точек:
После изменения y в точке 3 на 8.0:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,0, 8,0)
Точка 4: (4,0, 16,0)
После изменения x в точке 3 на 3.5:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,5, 8,0)
Точка 4: (4,0, 16,0)
```

```
После добавления и удаления точек:
После добавления x в точке 3 на 3.5:
Область определения: [0,0, 4,0]
Количество точек: 6
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,0, 3,5)
Точка 4: (3,5, 8,0)
Точка 5: (4,0, 16,0)
После удаления точки с индексом 3:
Область определения: [0,0, 4,0]
Количество точек: 5
Точка 0: (0,0, 1,0)
Точка 1: (1,0, 3,0)
Точка 2: (2,0, 4,0)
Точка 3: (3,5, 8,0)
Точка 4: (4,0, 16,0)
```